Complexity

Name	Complexity
Selection sort	All case: $O(n^2)$
Bubble Sort	All case: $O(n^2)$
Shell sort	**Worst:** $O(n^{3/2})$
Early-termination Bubble sort	**Best:** $O(n)$ **Worst:** $O(n^2)$ **Avg:** $O(n^2)$
Insertion sort	**Best:** $O(n)$ **Worst:** $O(n^2)$ **Avg:** $O(n^2)$
Merge sort	All case: $O(nlogn)$
Quick sort	**Best:** $O(nlogn)$ **Worst:** $O(n^2)$ **Avg:** $O(1.3999nlogn)$ Avg case 39% faster than `merge sort`
Heap sort	**Build + Deque** $O(nlogn) + O(n) = O(nlogn)$
DFS, BFS	**Adj Matrix:** $\Theta(\lvert V^2 \rvert)$ **Adj List:** $\Theta(\lvert V\rvert + \lvert E\rvert)$
Binary search	**Best:** $O(1)$ **Worst:** $O(log(n))$ If the array `is sorted` or has $O(1)$ access to any positon

BST Search	**Best:** $O(1)$ **Average:** $O(logn)$ **Worst:** $O(n)$ If it's a stick, that's $h = n - 1$
Search in sorted list	**Merge sort + binary search** $O(nlogn) + O(logn) = O(nlogn)$ Worth to presort when: $k >= \frac{nlogn}{n/2 - logn}$
AVL trees	**Height:** $h <= 1.4404log(n+2) - 1.3277$ **Search** and **Insert:** $O(logn)$ **Delete:** $O(logn)$ **Rebalance (rotations):** $O(logn)$ **Disadvantages:** Rotations are frequent and implementation is complex
2-3 trees	**Height:** $log_3(n+1) - 1 <= h <= log(n+1) - 1$ **SEARCH, INSERT, DELETE:** $O(logn)$ **Rebalancing:** < $O(logn)$
Heaps	**Height:** $h = logn$ **Repair:** $O(logn)$ **C(n)** = $O(n)$ **Building:** $O(nlogn)$
Distribution sorting	Worst case: $O(n)$ if $n > n_{max}$ Space: $O(n) + O(n_{max})$
	List - Insert: $O(1)$ - Delete: $O(n)$ - Search: $O(n)$ **Balance Tree** - Insert: $O(logn)$

Hash tables	- Delete: $O(logn)$ - Search: $O(logn)$ **Array** - Insert: $O(logn)$ - Delete: $O(logn)$ - Search: $O(logn)$ **Open Hashing** - Insert: $O(1)$ - Delete: propotional to list's length - Search: propotional to list's length - Average: $O(1)$
Prim's algorithm	**Min-heap + Adjacency list** - $O(\|E\|log\|V\|)$ **Fibonacci heap + adjacency list** - $O(\|E\| + \|V\|log\|V\|)$
Kruskal's algorithm	If there are edges: - $O(\|E\|log\|E\|)$ If there is no edge: - $O(\|E\|log\|V\|)$
Dijkstra's Algorithm	Min-heap: - $\Theta(\|E\|log\|V\|)$
Huffman's Code	Min-heap: - $\Theta(nlogn)$ Sorted by probabilities: - $\Theta(n)$
Edit distance	Worse and Average: - Construction: $\Theta(nm)$ where n and m is length of 2 strings. - Backtrace complexity: $\Theta(m+n)$

`Knapsnack problem`	`Build table:` $\Theta(nW)$ `Backtrace complexity:` $\Theta(n+W)$
`Warshall's algorithm`	`Complexity:` $\Theta(n^3)$ `Space:` $\Theta(n^2)$ `DFS & BFS:` $\Theta(n^2+nm)$ `for n,` $\Theta(n+m)$ `for 1` `If not many:` $\Theta(n^2)$

- **Stack**:
  - Add and remove at front.
    - [a] add b => [b,a]. remove b => [a]
- All `log` , `ln` has the same speed
- (n-1)! < n!
- When rotating for AVL trees. If one element changes its position to the right. We say R(that element). So draw first and then write rotation direction.